Decontamination from Black Virus Using Parallel Strategy

by

Yichao Lin

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the Master degree in
Computer Science

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

In this thesis, the problem of decontaminating networks from *black virus* (BVs) using parallel strategy with a team of system mobile agents (the BVD problem) is studied. The BV is a harmful process whose initial location is unknown a priori. It destroys any agent arriving at the network site where it resides, and once triggered, it spreads to all the neighboring sites, i.e, its clones, thus increasing its presence in the network. In order to permanently remove any presence of the BV with as less execution time as possible and minimum number of site infections (and thus casualties), we propose parallel strategy to decontaminate the BVs: instead of exploring the network step by step we employ a group of agents who follow the same protocol to explore the network at the same time, thus dramatically reducing the time needed in the exploration phase and minimizing the casualties. Different protocols are proposed in meshes, tori, and chordal rings following the monotonicity principle. Then we analyze the cost of all our solutions and compare to the asynchronous BV decontamination. Finally conclusion marks are presented and future researches are proposed.

Acknowledgements

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

A distributed system is a group of computational entities cooperating with each to achieve one or more tasks. This thesis deals with distributed computing by mobile agents in network. More specifically, we deal with the problem of deploying a group of mobile agents who follow the same protocol to explore the network and decontaminate the dangerous virus (called Black Virus) present on the network nodes. In this chapter, the motivations of the problem are provided, following is a brief summary of the contributions.Finally, an overview of the organization of the thesis is presented.

1.1 Problem and Motivation

Mobile agents are widely used in distributed and network systems while the applications of them can cause some security issues, thus threatening to the network: A contaminated or infected host can destroy working agents for various malicious purposes; A malicious agent can contaminate or infect other computer nodes so they become malfunctional or crash. The harmful hosts, often called *Black Holes* trigger the problem called *Black Hole Search* (BHS), the focus of which is to locate their positions since it is statics. This problem has been studied in many variants. For example, different topologies and different settings

(synchronous and asynchronous). The harmful agents trigger the problem called *Intruder Capture* (IC). Its main focus is to deploy a group of mobile agents to capture a extraneous mobile agent (the intruder) who moves arbitrarily fast through the network and infects the visiting sites. Also it has been investigated in a variety of topologies. More detailed literature review will be provided in Chapter 2. Note that BH is static and only damage the agents reaching it without leaving any detectable trace. Intruder is mobile and harmful to the network nodes but does not cost any harm to other system agents. A new harmful presence called *black virus* BV has been initially introduced by Cai et al. in[18]. It is a dangerous process resides at an unknown site in a network and destroys any upcoming agents, but unlike the BH, the node where the original BV resides thus become clean. At the same time, the original BV multiplies (called clones) and spread to all neighbouring nodes, thus increasing its number,and damage in the network. A BV is destroyed when it moves to a site where there is already an agent. Based on this harmful presence, a new problem called Black Virus Decontamination(BVD) is presented by Cai et al., the main focus of which is to use a group of system agents to permanently remove any presence of the BV from the network. A protocol defining the actions of the agents solves the BVD problem if at least one agent survives and the network is free of BVs. Also, a desirable property of a decontamination protocol is that the nodes which have been explored or cleaned by mobile agents are not be recontaminated by the BV spreading. A solution protocol with such a property will be called *monotone*. see[16]. Some important cost measure is the number of node infections by the BVs (casualties); size of the team, i.e, the number of agents employed by the solution, the time needed by the solution. Solutions in which the agents explore the network's node in sequence have been proposed in [18], [1] and [19]. The size of the team is minimum in [18, 19] and the number of site infections is also minimum in such case, i.e., exploring the network nodes in sequence but the units of time that these protocol (including the protocol in chordal rings) cost is usually several times as much as the total number of the network's nodes (assuming in synchronous setting). Now

we are interested in the solution using parallel strategy where we deploy a larger number of mobile agents following the same protocol to decontaminate the network in the exploring phase with the goal to minimize the *total working time* (TWT) which is calculated by multiplying the number of agents and the total execution time and also the casualties.

1.2 Our Contribution

1. In this thesis, we propose parallel strategy to solve the BVD problem. It is the first attempt to deal with this issue in a parallel way. Agents are not allowed to communicate with each other unless they are in the same network node so the protocol should enable the agents in different nodes to move properly, i.e, the route of every agent is different but they are served to explore the network; when a BV is triggered, other agents should bypass the new-formed BVs... We give simple but efficient solution to deal with this problem with acceptable cost. Also we give the size of the exploring team which is minimum to guarantee both the TWT and the casualties we reach.

2. The BVD problem is investigated for three important topologies: *meshes*, *tori*, *chordal rings*. All the protocol are optimal both in term of TWT and casualties. We compare our solution with [18] and [1] in which the exploring route is in sequence and the result is that our solution is better than that of them in both TWT and casualties. One should be point out that in chordal ring especially, the more complicated the chordal ring becomes, the more TWT that we save comparing to [1].

1.3 Thesis Organization

The thesis is organized as followed:

Chapter 2 contains a literature review on related problems. We begin by reviewing

the Black Hole Search and Intruder Capture problem, and then focus on the solution of BVD problem where the mobile agents explore the network in sequence, the issue has been studies in different topologies: two-dimensional grids, three-dimensional grids, tori, chordal rings, hypercubes and arbitrary network. Also, the variant of this problem, which is decontamination of an arbitrary network from multiple black virus is also reviewed. We also present our assumption and the topologies (meshes, tori, chordal rings).

Chapter 3 introduces terminology, definitions and model for the BVD problem used in the rest of the thesis. Also we describe the high level ideas that serve as the basic of all our solutions. Since monontone is the necessary condition for spread optimality, we go through the principle and finally make a conclusion.

Chapter 4 focus on the BVD problem for two simple classes of network topologies: *meshes* and *tori*. For each kind of network, an optimal algorithm in terms of casualties and TWT is developed. Complexity analysis in terms of casualty and TWT are performed and obtained. Some comparison and analysis are also made between our solution and [18] and the result shows that our solution is better.

Chapter 5 presents the BVD problem for the chordal ring topology. In this chapter, we introduce the *Three Jump Notifying Technique* (TJNT) to manipulate each mobile agent efficiently go through their route in the exploring phase and avoid any new-formed BV after the original BV is triggered. Based on this technique, we develop the parallel strategy for the mobile agents to decontaminate the choral ring from BV. Complexity analysis in terms of casualty and TWT are performed. Finally some comparison and analysis are made between our solution and [1] and the result shows that our solution performs better.

Chapter 6 summaries the main conclusion of our work and present some open problems and future work.

Chapter 2

Literature Review

Mobile agents have been widely used in the field of distributed computing due to their features especially the mobility which allow them to migrate between computers at any time during their execution. A group of agents can be used to perform a various tasks, for example, network exploration, maintenance, and etc. However, the introduction of mobile agent tend to cause security problem, thus threatening the network. Various security issues and solution algorithms have been proposed by Flocchini and Santoro in [25]. Generally, the threaten that the mobile agents cause are divided into two categories: in first case,the malicious agents can cause network nodes malfunction or crash by contaminating or infecting them (the harmful agent); in second case, the contaminated or infected hosts can destroy working agent for various malicious purposes (the harmful hosts). These two threaten trigger two problems: Black Hole Search (BHS) and Intruder Capture (IC) which will be introduced in the following sections. Then we review the BVD problem which deal with the decontamination of a harmful presence which cause the network node malfunction but leaves the network node clean when it is triggered and spreads to all its neighbouring nodes, thus increases it presences. In the section introducing BVD problem, we present the the abilities of mobile agents that has been proposed and different decontamination strategies based on different strategies.

2.1 Black Hole Search, BHS

The BHS problem assumes there is a BH or multiple static BHs residing at certain network nodes and will destroy any upcoming agents without leaving any detectable trace.The task is to use a team of agent to locate the black hole(s) and is completed when at least one agent survives and reports the location(s) of the black hole(s). Note that the solution is based on graph exploration and the goal can be reached totally depending on the sacrifice of some agents. In [26], Das et al. considered a model for unknown environment with dispersed agents under the weakest possible setting, many exploration models and works were included in this article. The BHS problem has been widely studies in various topologies and settings: the timing is synchronous or asynchronous; the number of black hole(s) is known or unknown. For example: by Chalopin [20, 21] in asynchronous rings and tori, Dobrev et al. [28] in arbitrary graph, [27] in anonymous ring and [29] in common interconnection networks...What is worth pointing out is that the number of BHs remains the same as it of the beginning, thus not causing harm to other sites of the network.

2.2 Intruder Capture

The IC problem assumes that there is an intruder moves with an arbitrary speed from node to node in the network and contaminate the sites it visits, the goal of which is to deploy a group of mobile agents to capture the intruder;the intruder is captured when it comes in contact with an agent. Note that the intruder does not cause any harm to the upcoming agents. It is equivalent to the problem of decontaminating a network contaminated by a virus while avoiding any recontamination. This problem is first introduced in [3] and has been widely investigated in a variety network topologies: trees [2, 3, 10], hypercubes[13], multi-dimensional grids[13], chordal rings[14] etc. The studies of arbitrary graph has been started in [4, 17]. Note that monotone is a critical principle in the solutions of IC problems.

2.3 Agent Capabilities

Different capacities granted to the mobile agents have an impart on solving the BHS problem,IC problem and also the BVD problem.now we discuss these capabilities in the following section.

Communication Mechanisms Mobile agents can communicate with each other only when they are in the same node in a network.Some essential communication methods have been studied in literature: whiteboard,tokens and time-out. In [8, 15, 22, 28], the whiteboard model is used, which is a storage space located at each node and agents arriving there are able to read and write. In the token model, (see [5, 15]), tokens are like memos of the agents which can be dropped off and picked by agents at nodes or edges. While the time-out mechanism can only be used in synchronous setting where each agent has a pre-determined amount of time. (see [6, 7, 9]).

Knowledge of the topology Different assumption of mobile agents' knowledge of the topology has an impact on solutions of some of the problems mentioned above, for example, the BHS problem. In [28], Dobrev et al. present three types of topological knowledge in an asynchronous arbitrary network and show the results of the BHS problem based on different setting of the topology knowledge.

Other capabilities In some studies, agents are endowed with the visibility, which mean that they can see whether or not their neighbouring nodes are clean or contaminated (see [11, 12]).They observe that the visibility assumption allow them to drastically decrease the time and move complexities in torus, chordal ring and hypercubes when dealing with IC problem. For example, in chordal ring $C_n\{d_1 = 1, d_2, ..., d_k\}$, the number of agents, the time and the moves required in local model are $(2d_k + 1)$, $3n - 4d_k - 1$, $4n - 6d_k - 1$ respectively, while in visibility model, they are $2d_k$, $\left\lceil \frac{n-2d_k}{2(d_k-d_{k-1})} \right\rceil$, $n - 2d_k$. In torus , the number of agent,

the time and the moves required are $2h+1$, $hk-2h$, $2hk-4h-1$ and in visibility model are $2h$, $\lceil \frac{k-2}{2} \rceil$, $hk-2h$ respectively. They also compare the complexity of both models in hypercubes, a algorithm requiring $\Theta\ (\frac{n}{\sqrt{logn}})$ number of agents and $O(n \log n)$ moves while the algorithm they propose in the visibility model requires $\frac{n}{2}$ agents and $O(n \log n)$ moves.

In [**?**], the concept of *k-hop visibility* is presented. The agents have the *full topologies* if each of them have a map in their memory of the entire network including the identities of the node and the labels of the edges. If a agent has *k-hop visibility*, then at a node v a agent can see the k-neighbourhood $N^{(v)}$ of v, including the node identities and the edge labels. Note that *Diam-hop* visibility is equivalent to full topological knowledge.

Another interesting capability of agents is cloning which is introduced in [12]. Cloning is the capacity for an agent to create copies of itself. In this paper, they also discuss how the combination of different capacities reaches different optimal strategy in IC problem in hypercube. For example, the strategy is both time and move-optimal when visibility and cloning are assumed or when cloning, local and synchronicity are assumed. But the time and move-optimal strategy can be obtained at the expense of increasing the number of agents. The last capability of agents be discussed is immunity which means that a node is immune from recontamination after an agent departs. Two kinds of immunity have been proposed:local and temporal. In local immunity, (see [23, 24], the immunity of a node depends on the state of its neighbouring nodes. More specifically, a node remains clean after the departure of an agent until more than half of its neighbours are contaminated. In the temporal immunity, a node is immune for a specific amount of time (t). The node remains clean until time expires and becomes recontaminated if at least one of its neighbours are contaminated. In models without immunity assumption, a node becomes recontaminated if it has at least one contaminated neighbours.

2.4 Black Virus Decontamination

2.4.1 Overview

The BVD problem is first introduced by Cai at al. in citeCai: A black virus is a extraneous harmful process endowed with capabilities for destruction and spreading. The location of the initial BV(s) is known a priori. Like a BH, a BV destroy any agents arriving at the network where it resides. When that happen, the clones of the original BV spread to all it neighbouring nodes and remain inactive until an agent arrives. The BVD problem is to permanently remove any BVs in the network using a team of mobile agents. They proposed that the only way to decontaminate a BV is to surround all its neighbouring nodes and send an agent to the BV node. In this case, the node where the original BV resides is clean and all its clones are destroyed by the guarding agents in its neighbouring nodes.They have presented different protocols in various topologies: q-grid, q-torus, hypercudes in [18] and arbitrary graph [19]. A basic idea of implementing the decontamination has also been propose by them assuming that the timing is asynchronous which divides the whole decontamination process into two part: '*shadowed exploration*' and '*surround and eliminate*'. In order to minimize the spread of the virus, they use a '*safe-exploration*' technique which is executed by at least two agents: the *Explorer Agent* and the *Leader Explorer Agent* who both residing at a safe node u at the beginning, for example,the homebase. The *Explorer Agent* moves to a node v to explore it and it needs to return to node u to report the node v is safe. The *Leader Explorer Agent* determines if the node v is safe or not by *Explorer Agent*'s arriving or a BV's arriving. If node v is safe, both of them move to node v. For the purpose of insuring monotonicity, at any point in time the already explored nodes must be protected so they are not be recontaminated again. After the BV is detected, the '*surround and eliminate*' begins. In this phase, some agents are employed to surround the new-formed BVs (the clones of the original BV) then some agents are sent to the clones to permanently destroy them. This is called the '*Four-step*

Cautious Walk' and is widely used in BVD problem with synchronous setting. Also, BVD problem in chordal ring has been discussed in [1].

2.4.2 BVD in different topologies

Protocols regarding to BVD problems in grid are BVD-2G and BVD-qG which deal with BVD problems in 2-dimensional grid (meshes) and q-dimensional grid respectively. BVD-2G performs a BV decontamination of a 2-dimensional grid of size n using $k = 7$ agents and 3 casualties, within at most $9n + O(1)$ moves and at most $3n$ time. While protocol BVD-qG performs a decontamination of a q-dimensional grid of size $d_1 \times d_2 ... \times d_q$ using $3q + 1$agents and at most $q + 1$ casualties, within at most $O(qn)$ moves and at most $\Theta(n)$ time. Algorithm to decontaminate the BV in a q-dimensional torus,called BVD-qT uses $4q$ agents with 2 casualties with at most $O(qn)$ moves and $\Theta(n)$. Protocol BVD-qH is to perform a BV decontamination of a q-hypercube using $2q$ agents and q casualties with at most $\mathrm{O}(n \log n)$ moves and $\Theta(n)$. In arbitrary graph (see [19]), two protocols are presented: GREEDY EXPLORATION and THRESHOLD EXPLORATION. In these two protocols, $\Delta + 1$ agents are needed and both of the protocols are worst-case optimal with respect to the team size. Though the protocols are described for a synchronous setting, they easily adapt to asynchronous ones with an additional $\mathrm{O}(n)$ moves for the coordinating activities. An advantage of these protocols is that the agents can use only local information to execute the protocol. Another interesting fact based on these two protocols is that both GREEDY ROOTED ORIENTATION and THRESHOLD ROOTED ORIENTATION produce an optimal acyclic orientation rooted in the homebase.

In [1], solution for BVD in chordal ring is discussed. In Alotaibi's thesis, she discuss solutions based on different kinds of chordal ring: double loops, triple loops, consecutive-chords rings and finally general chordal ring. In double loops, she proposed three strategies in elimination phase and the upper bound of moves is $4n - 7$ in the whole protocol and

a maximum of 12 agents are employed.In triple loops, she discusses two classes of chordal ring: $C_n(1, p, k)$ and $C_n(1, k-1, k)$. In any triple loop $C_n(1, p, k)$, a maximum of $5n - 6k + 22$ moves and 24 agents are needed for the decontamination while in any triple loop $C_n(1, k-1, k)$, a maximum of $5n - 7k + 22$ moves and 19 agents are needed. Finally in the consecutive-chords ring, a maximum of $(k+2)n - 2k - 3$ moves and $4k + 1$ agents are needed. She described the decontamination strategies in synchronous setting but only with a cost of $O(n)$ moves can the strategies be used in asynchronous setting.

Chapter 3

Definitions and Terminology

3.1 Model

3.1.1 Network, Agent, Black Virus

Network The environment in which mobile agents operate is a network modelled as simple undirected connected graph with $n = |V|$ nodes (or sites) and $m = |E|$ edges (or links). We denote by $E(v) \subseteq E$ the set of edges incident on $v \in V$, by $d(v) = |E(v)|$ its degree, and by$\Delta(G)$ (or simply Δ) the maximum degree of G. Each node v in the graph has a distinct $id(v)$. The links incident to a node are labelled with distinct port numbers. The labelling mechanism could be totally arbitrary among different nodes; without loss of generality, we assume the link labelling for node v is represented by set $l_v = 1, 2, 3, ..., d(v)$.

Agent A group of mobile agents are employed to decontaminate the network. The agent is modelled as a computational entity moving from a node to neighbouring node. More than one agents can be at the same node at the same time. Communication among agents occurs at this time; there are no a priori restrictions on the amount of exchanged information. In our thesis, we employ two groups of agents (the exploring group and the shadowing group) operating the same protocol and we assume that all the agent?s moves follow the same

clock. Agents in the exploring group Also, agents are endowed with 1-hop visibility which means at a node v, it can see the labels of the edges incident to it and the identities of all its 1-hop neighbours. We do not guarantee other capability introduced in Chapter2 to the agents in our protocols.

Black Virus In G there is a node infected by a black virus (BV) which is a harmful process endowed with reactive capabilities for destruction and spreading. The location of the BV is not known at the beginning. It is not only harmful to the node where it resides but also to any agent arriving at that node. In fact, a BV destroy any agent arriving at the network site where it resides, just like the black hole. Instead of remaining static as the black hole, the BV will spread to all the neighbouring sites leaving the current node clean. The clones can have the same harmful capabilities of the original BVs (fertile) or unable to produce further clones(sterile). In this thesis, we only consider the situation of fertile clones but actually the protocol can be simply modified to adapt to the sterile situation. A BV will be destroyed if and only if the BV arrive at a node where there is already an agent. Thus, the only way to eliminate the BV is to surround it completely and let an agent attack it. In such situation, the attacking agent will be destroyed while the clones of the original BV will be permanently eliminated by the agents residing the neighbouring nodes of the original node. We assume that at the same node, multiple BVs (clone or original) are merged. More precisely, at any time, there is at most one BV at each node. Another important assumption is that when a BV and an agent arrive at an empty at the same time, the BV dies and the agent survive remaining unharmed.

Summarizing, there are five possible situations when an agent arrive at a node v:

- agents arrive at a node which is empty or contains other agents, they can communicate with each other and the node v is clean.
- agents arrive at a node which contains a BV, the agent dies and the clones of the BV (BVC) spread to all the neighbours of v leaving node v clean.

- A BVC arrives at a node which is empty or there is already a BV: the node becomes/stays contaminated; it merges with other BVs.
- BVCs arrive at a node v which contains one or more agents, the BVCs are destroyed but the agents are unharmed.
- A BVC and an agent arrive at an empty node at the same time, the BVC dies while the agent remains unharmed.

3.1.2 Problem, Cost

The BLACK VIRUS DECONTAMINATION (BVD) problem is to permanently remove the BV, and its clones from the network using a team of mobile agents starting from a given node, called home base(HB). The solutions where the agents explore the network sequentially have been proposed in some classes of topologies. chordal rings, hypercubes and arbitrary graph. In this thesis, we are interested in parallel strategies in BVD problem: instead of exploring the network in sequence, we explore it in parallel; in chordal ring, we also propose a parallel solution to surround the clones of the original BV. In this thesis, the efficiency measurements we have are: *spread* of BV (also measures the number of agents *casualties*; the *size* of the team, i.e, the number of agents employed by the solution; total working time(TWT) (calculated by multiplying the *size* of the team and the time cost by the solution. Note that TWT does not contains any practical meaning but exist only as a measurement. We propose TWT to compare more fairly the time of two protocols when the number of agents is different.

3.1.3 Monotone, Synchrony

A desirable property of a decontamination protocol is to prevent the nodes which been explored or cleaned by mobile agent from being recontaminated which will occur if the

clones of the BV are able to move to a explored node in absence of agent. A BVD protocol with such property is called monotone. Monotone property is the necessary condition for spread optimality.

Asynchrony refers to the execution timing of agent movement and computations. The timing can be *Synchronous* or*Asynchronous*. When the timing is synchronous, there is a global clock indicating discrete time unit; it takes one unit for each movement (by agent or BV); computing and processing is negligible. When we have asynchronous agents, there is no global clock, and the duration of any activity (e.g., processing, communication, moving) by the agents, the BV, and its clones is finite but unpredictable. In this thesis, all our protocols work in synchronous setting.

3.2 General strategy

Following the solution in sequential case, we decomposed the BVD process into two separate phase: *Shadowed Exploration* and *Surrounding and Elimination*. The task of the first phase is to locate the BV and the second phase is to decontaminate the BV and its clones. Apart from these two basic phase, we have initialization which is to deploy the agents properly at the beginning of executing the protocol because we explore the network in parallel, and the arrangement of the agents is crucial to successfully execute the protocol.

Phase1: Shadowed Exploring Agents employed are divided into two group: shadowing group and exploring group, and the number of agents in two group is the same. For convenience, we call the agent in the shadowing group the shadow agent (SA and those in the exploring group the exploring agent EA (one EA is accompanied by one SA. As the name indicated, agents in exploring group explore the network and the agents in shadowing group follow the agents protecting the node which have been explored. More precisely, at T_1, EA moves to node v; at T_2, SA moves to node v and EA moves to node u supposing that node v is clean.

Phase2: Surrounding and Elimination In this phase, since we already know the position of the BV, we employ agents to surround all the neighbours of the BVCs. Once all the agent arrive the proper positions(all the neighbours of the BVCs are guarded), we employ another group of agents (the number of them is equal to the number of BVCs) to move to the BVCs, thus permanently destroy them. Usually, some of the agents moving to the neighbours of the BVCs are from the shadowing group and exploring group because in this way we can save the number of agents used in the whole protocol. Note that not all the agents are informed when the BV is detected. More specifically, only the agents who receives the clones know the existence of the BV, and other agents keep moving in the network. In some simple topologies, such as meshes and torus, the second phase begins when the BV is detected since the number of agents are enough to proceed the second phase. In some more complicated topologies, for example, the chordal ring which we discuss later, we take some other measures to call back enough number of agents to finish the second phase.

3.3 Conclusion

In this Chapter, we presented the model of our problem and also some important terminologies. Also we described a general strategy for our problem depending on particular setting: synchronous timing, parallel strategy... In the next chapter, we discuss the parallel strategies in BVD problem in two simple topologies: meshes and torus.

Chapter 4

Parallel Black Virus Decontamination in Mesh and Torus

Chapter 5

Parallel Black Virus Decontamination in Chordal Ring

5.1 Introduction

In this chapter, we discuss parallel strategy on BVD problem in chordal ring. A chordal ring is a circulant graph with $d_1 = 1$, i.e., it is an augmented ring, and will be denoted by $C_n(1, d_2, ..., d_k)$. More specifically, in chordal ring each node is directly connected to the nodes at distance d_i by additional links called chords. The link connecting two nodes is labeled by the distance that separates these two nodes on the ring. For convenience, if we say the agents or the clones move along chord i, then they actually move along d_i. If we say the agents or the clones move i, then they actually move along chord d_x with its length equal to i. Let us donated by d the half degree of the chordal ring (for example, for the chordal ring structure $C_n(1, 2, 4, 5)$, $d = 4$), by l the length of the longest chord of the chordal ring (for example, for the chordal ring structure $C_n(1, 2, 4, 5)$, $l = 5$). In order to simplify the process, we assume that all the nodes in the network are marked with a number: the starting point is marked 0, then the second node is marked 1... Our goal is to minimize the time to complete the whole decontamination process and at the same time

the casualties. In order to do that, we propose a parallel strategy for decontaminating the chordal ring. It is the first attempt to deal with this issue in a parallel way. Agents are not allowed to communicate with each other unless they are in the same node so the protocol should enable agents in different nodes to move properly. That is, the route of every agent is different but they are served to explore the network; when a BV is triggered, other agents should bypass the new-formed BVs. We give simple but efficient solution to deal with the problem with acceptable cost. In the elimination phase, we are interested in executing it also parallelly. But in this case, an tricky situation might happen: supposing that the sites of the two clones are connected, in this case, after these two clones are triggered, one of their clones spread to another sites and since the agents sent to destroy them die, these two sites are empty when the second round clones arrive, which make our decontamination invalid. In order to solve this problem, we make an assumption when we talk about the parallel strategy in elimination phase in chordal rings: when a BV is triggered at T_i, it takes negligible time for its clones to reach all its neighbours, for example, at T_i. Since time cost in the elimination phase is negligible comparing to the exploring phase if n is large enough, it would be much simple to execute the elimination sequentially. That is, instead of destroy all the clones at the same time, we decontaminate them on by one. Also, in the elimination phase, in order to save the number of agents, we chase all the $Keep\,Moving$ agents (explained in section Surrounding and Elimination) back. It is obvious that we may cost some time here, if the execution time is of great important to us and we care less the number of agents we employ, then we may carry enough agent at the beginning and start the elimination phase right after the exploring phase. In conclusion, In the exploring phase, we have one strategy, but in the elimination phase, we provide several strategies: 1) chasing back the $Keep\,Moving$ agents and decontaminating the BVs sequentially 2) chasing back the $Keep\,Moving$ agents and decontaminating the BVs parallelly 3) start to decontaminating the BVs right after the exploring phase and do it parallelly 4) start to decontaminating the BVs right after the exploring phase and do it parallelly. The cost of

each strategy including time, number of agents, casualties would be analyzed in the section Analysis and Comparing.

5.2 Shadowed Exploration

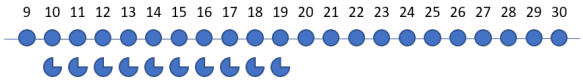
Initialization

The chordal ring is a complete symmetrical structure, so we can randomly choose a node x_0 as the start node. Initially, we place $2d$ agents at node 0, 1, ..., $2d-1$ (first round). Agents residing in nodes from 0 to $d-1$ are in shadowing group, while from d to $2d-1$ are in exploring group. If none of the agent is destroyed, we then place d agents at nodes 0,1, ..., $d-1$ (second round). If not, we can easily know the location of the BV and employ agents to surround the new formed BVs, then destroy them permanently. Only the agents employed in the first round move in the exploring phase. The agents employed in the second round remain dormant, guarding the nodes to guarantee the monotone.

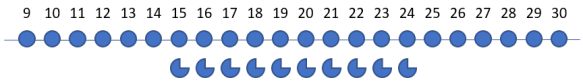
Route of the agent in exploring phase

We separate the time of exploring phase into two part: moving time and notice time. In the whole phase of exploring phase, they are arranged as below: T_{move_1}, T_{notice_1}, $T_{notice\_1'}$, $T_{notice\_1''}$, T_{move_2}, T_{notice_2}, $T_{notice\_2'}$, $T_{notice\_2''}$ ...More specifically, every cycle contains one unit of time for moving and three units of time for notice. We discuss why we arrange the time as above and what exactly the agents do in the notice time later.

In chord ring $C_n(1, d_2, \ldots, d_k)$, all the agents in the array move along their longest chord d_k in T_{move_i}. That is, agents move along d_k in $T = 1 + 4t$ ($t \in \mathbb{N}$). An example of how agents move in chordal ring $C_n(1, 2, 4, 5)$ at T_{move_i} in exploring phase is shown in Fig.5.1.



(a) Arrangement of agents at T_{move_i}



(b) Arrangement of agents at T_{move_i+1}

Figure 5.1: Arrangement of agents when moving

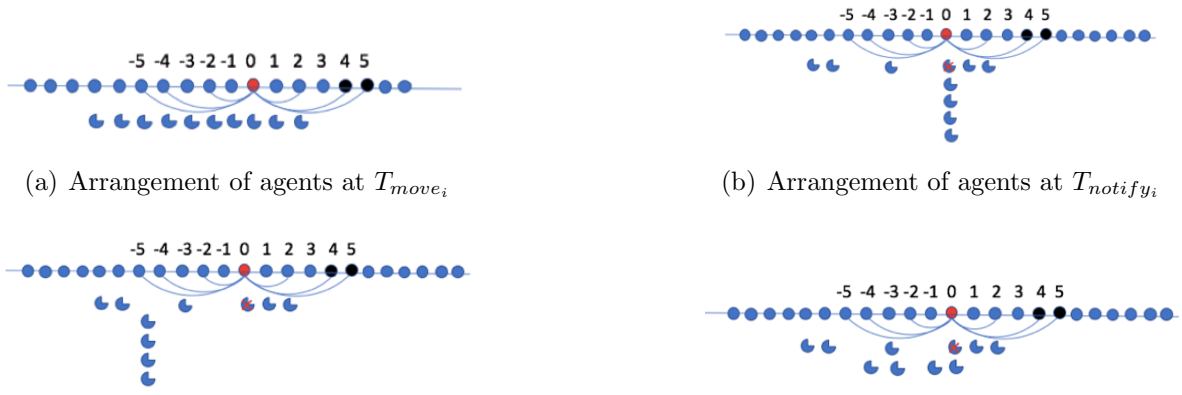
Three Jump Notifying Technique In the strategy of BV decontamination exploring the network sequentially, two agents explore the graph (exploring agent and leader agent) and they follow the casual walk. That is, exploring agent moves to the next node in its route, and if the node is safe, it moves back to the leader agent, and then move forward to that safe node together. In this case, if the leader agent does not meet the exploring agent in next T after the exploring agent moves forward, the leader agent learns that the original BV resides in the next node so it stop moving.

But in our model, we employ $2d$ agents in the exploring phase, if they are not properly informed when one agent is destroyed by BV, in their next step of moving, some of them may be destroyed by the new formed BVs. In order to avoid the casualties, we propose $Three\ Jump\ Notifying\ Technique$ to properly notify the agents who will move to the new formed BVs.

To explain it more clearly, let us take the node where the original BV resides as the original of the one-dimensional coordinate system. In a chordal ring $C_n(1, d_2, \ldots, d_k)$, if the original BV is triggered, the clones of it spread to nodes whose coordinates are $-d_k$, $-d_{k-1}$, $\ldots$, $-d_2$, -1, 1, d_2, $\ldots$, d_{k-1}, d_k. Obviously the nodes whose coordinates are 1, d_2, $\ldots$, d_{k-1}, d_k may become the BV (Possible BV Nodes) now, our goal is to notify the agents which will move to these nodes ($Risky\ Agent$) to stop. The coordinates of $Risky\ Agent$

(RAs) are $1-d_k$, d_2-d_k, ..., $d_{k-1}-d_k$, d_k-d_k (which is exactly the coordinate of the original BV) respectively. It is obvious that not all of the nodes from 1 to d_k become BV nodes after the triggering because there might be some agents already there, but since notifying all the *Risky Agents* does not add more cost comparing to notifying some of them, in our strategy, we notify all of the RAs. Let us donate one of the possible BV nodes by d_i, and in our $Three\,Jump\,Notifying\,Technique$, agent residing in node $-d_i$ ($Notification\,Agent$) is employed to notify the agent residing in node $-d_k+|d_i|$ (RA) who will move to the BV node.

We show that $Notification\,Agent$ (NA) are able to meet RA in three steps: $-d_i \xrightarrow{move|d_i|} 0 \xrightarrow{move\,along\,chord\,k} -d_k \xrightarrow{move\,|d_i|} -d_k+|d_i|$. In this case, the notifying route of the NA whose coordinate is $-d_k$ is $-d_k \rightarrow 0 \rightarrow -d_k \rightarrow 0$. We would make some modification of this agents route in the $Surrounding\,and\,Elimination$, but now let us assume it still follow the route above. The whole process of $Three\,Jump\,Notifying\,Technique$ in chordal ring $C_n(1,2,4,5)$ is shown in Fig.5.2.



(a) Arrangement of agents at T_{move_i}

(b) Arrangement of agents at T_{notify_i}

(c) Arrangement of agents at $T_{notify'_i}$

(d) Arrangement of agents at $T_{notify''_i}$

Figure 5.2: The whole process of the $Three\,Jump\,Notifying\,Technique$ in chordal ring $C_n(1,2,4,5)$

If the original BV residing in the red node, then once an agent moves to it, the agent and the BV are destroyed but the clones of the BV spread to all its neighbours. According

to our technique, nodes whose coordinates are 1, 2, 4, 5 become $Possible\ BV\ Nodes$; agents residing in nodes -4, -3, -1, 0 are the RAs; agents residing in nodes -5, -4, -2, -1 are the NAs. The routes for agents residing in nodes -5, -4, -2, -1 are $-5\rightarrow 0\rightarrow -5\rightarrow 0$; $-4\rightarrow 0\rightarrow -5\rightarrow -1$; $-2\rightarrow 0\rightarrow -5\rightarrow -3$; $-1\rightarrow 0\rightarrow -5\rightarrow -4$ respectively.

Safe Exploring with Three Jump Notifying Technique

After the initialization, agents employed in the first round move as the route we introduced in "Route of the agent in exploring phase". When one of the agents is destroyed at T_{move_i}, then $Three\ Jump\ Notifying\ Technique$ begins. In "Route of the agent in exploring phase", we say that in the exploring phase every cycle contains one unit of time for moving and three units of time for notifying. Actually, the three units of time for notifying are reserved for $Three\ Jump\ Notifying\ Technique$, even though it only executes when one of the agents is destroyed. In another word, before encountering a BV, all the agents just stay where they are in the notifying time. After executing the $Three\ Jump\ Notifying\ Technique$, the NA moves back to where they are before the notification. For example, in the example in $Three\ Jump\ Notifying\ Technique$, NA residing in node -1 moves back to node -4 following the reverse route in the notifying phase which is $-1\rightarrow -5\rightarrow 0\rightarrow -4$.

5.3 Surrounding and Elimination

It is obvious that not all the agents are informed to stop so we call the agents continuing to move the $KeepMoving$ agents. In this section, we introduce the process of eliminating the BVs after the original BV is triggered. For the purpose of saving the number of agents, we prefer to chase the $Keep\ Moving$ agents, but it is not necessary to complete the process especially when you care most the execution time, you can carry enough number of agents so you can proceed the $Surrounding\ and\ Elimination$ immediately. The number of agents

that should be carried in order to successfully proceed the *Surrounding and Elimination* is discussed later in section Analysis and Comparing. Now we introduce how to chase the $KeepMoving$ agents.

5.3.1 Notifying Moving Agents

Overview of the Notifying Moving Agents

When the *Shadowed Exploring* ends, it is possible that some of the agents in the array are not informed and do not realize the existence of the BV, so they keep moving following the routes in *Shadowed Exploring* phase but it is obvious that they would not encounter any BV. In order to reduce waste, we employ the agent who receives the clone from chord d_k (*Coordinate Agent*) to notify the other $Keep\,Moving$ agents to move back to their position when the BV is triggered. Now we introduce how we choose the *Coordinate Agent*(CA)and how the CA notifies the other agents.

The Process of the Notifying Phase of the Coordination Agent

We can see that the relative position of agents does not change when they keep moving along the longest chord. For example, agents residing in nodes 1, 5, 10 (noted as $A1$, $A5$, $A10$) move along the longest chord and then $A5$ moves forward for one step. The relative position of them would be exactly the same as the situation where all of them remain dormant except $A5$ moves forward for one step. Now we discuss how the CA notify all the $Keep\,Moving$ agents assuming they are dormant and then make some modification to fit the scenario where the CA notifies the $Keep\,Moving$ agents. The problem we need to solve is that given a range within which the agents are in and a CA located in this range, let the CA to notify all the agents in this range. In a chord ring $C_n(1, d_2, \ldots, d_k)$, the CA sets a $Notification\,Window$ using the modular arithmetic. Let us assume the number of the node where the CA resides in is x, the number of the nodes where should be marked

the $Beginning\,Flag$ and the $End\,Flag$ are y and z.

The relations between x, y, z should be as follow:

- y is the biggest number which satisfies that it is smaller than or equal to x and that y mod d_k =0;
- z is the smallest number which satisfies that it is bigger than x and that z mod d_k = d_{k-1}.

When the agent is chosen to be the CA, it computes its $Notification\,Window$. When we mention marking a flag, it does not mean that the agent has to move to the node to do that but only needs to remember the positions of the two flags in its memory. Also, after the position of the $Notification\,Window$ is set, it remains stable. More specifically, when the CA moves, he does not set another $Notification\,Window$ using its new position. The CA moves step by step to every possible position and notifies the agents to go back if there is one until it realizes that it just passes the $End\,Flag$. After that, he moves along the longest chord anticlockwise to the node marked a $Beginning\,Flag$ and continues moving again step by step to notify agents until it arrives its relative departing node. For example, if the CA starts to notify other from node x, then its relative departing node is $x' = x + t \times d_k$ ($t \in \mathbb{N}$).

We make some modifications to let the solution fit the real scenario where the agents move along the longest chord:

- The Beginning Flag and the End Flag move along the longest chord also to keep the relative position the same.
- When one agent $A1$ moves to a node where there is an agent $A2$ knowing the position of the original BV, $A1$ would be informed and directly moves along the longest chord to its own position.

- Let us assume that the time when the original BV is triggered is T_{move_i} ($T_{trigger}$), then the CA should remember the $T_{trigger}$ and informs the agents he encounters of it. The agent $A1$ who encounters the CA should remember the time when they encounter (T_{notify_now}) and stop moving until next T_{move} when it will meet another agent $A2$. Then $A1$ moves along d_k anticlockwise for $T_{move\_now} - T_{trigger}$ times while $A2$ moves for $T_{move\_now} - T_{trigger} + 1$ times.

- When arriving its relative departing position at T_{move_a}, the CA knows that it has finished the task and moves along d_k for $T_{move\_a} - T_{trigger} + 1$ times to its position when the original BV is triggered.

- Let us assume that the time when the BV is triggered is T_{move_i}, the CA starts to move step by step to notify other agents after T_{move_i+2}, because we want to ensure the security of the CA. If it starts to notify other agents at T_{move_i+1}, it might encounter a BV. Also, it should be ensured that the $Keep\,Moving$ agents in the exploring group are in the $Notification\,Window$ of the CA, or the CA would never encounter them. We would talk about how to ensure this in next section.

Election of the Coordination Agent

We choose the agent who receives a BV clone from its chord d_k to be the CA, which means when an agent receives a BV clone from its longest chord, then it realizes that it is chosen as the CA. We know that, the notifying phase starts from T_{move_i+2} assuming the time when the BV is triggered is T_{move_i}, which means the notifying phase begins only after all the $Keep\,Moving$ agents move twice. At T_{move_i+2}, all the $Keep\,Moving$ agents are in a $Notification\,Window$ from nodes $d_k \times (move\_i + 1)\,to\,d_k \times (move\_i + 1) + d_k - 1$, and let us donate it by $Initial\,Notification\,Window$. The CA would directly move to any node in $Initial\,Notification\,Window$. Now we propose three kinds of routes for the CA to move to his destination:

The coordinates of the positions where the clones spread are: $x - d_k$ (which is the coordinate of the CA), $x - d_{k-1}$, $x - d_{k-2}$, $\ldots$, $x - 1$, $x + 1$, $x + d_2$, $\ldots$, $x + d_{k-1}$, $x + d_k$ supposing the coordinate of the original BV is x and we are in a chordal ring $C_n(1, d_2, \ldots, d_k)$. Now we describe three scenarios:

- Scenario 1: The last agent of the *Exploring Group* is destroyed by the BV and the positions of the clones satisfy: $x - d_{k-1} + d_k = x + 1$, $x - d_{k-2} + d_k = x + d_2$, $\ldots$, $x - 1 + d_k = x + d_{k-1}$.

- Scenario 2: The last agent of the *Exploring Group* is destroyed by the BV and the at least one pair of the positions of the clones does not satisfy: $x - d_{k-1} + d_k = x + 1$, $x - d_{k-2} + d_k = x + d_2$, $\ldots$, $x - 1 + d_k = x + d_{k-1}$.

- Scenario 3: One of the agents in the *Exploring Group* except the last agent is destroyed by the BV.

In scenario 1, the CA needs to move for 5 steps to reach its destination while in the other two scenarios, it only needs to move for 4 steps to arrive the destination. Now we propose the route for each scenario.

- For CA in scenario 1: Let us donate by y the coordinate of the node in the $Notification\,Window$ set by the original BV which does not receive any clone and his left neighbour receives a clone (the coordinate of it is $y - 1$). The CA first moves to the original BV, then to node $y - 1$, finally to y. After that it only need to move along the chord d_k for twice to reach its destination.

- For CA in scenario 2: There is at least one pair of the positions of the clones does not satisfy the equations so there should be one node (assuming its coordinate is z) who receives a clone from the original BV but node $z + d_k$ is empty. The route now for the CA is first move to the BV, then to node z, and then moves along the chord d_k for twice to reach its destination.

- For CA in scenario 3: The CA here simply need to move for one step to its right neighbour and move along the chord d_k for three times to reach its destination.

In any case, the CA can reach its destination within 6 unit of time which is required for the $Keep\,Moving$ agents to move to the $Initial\,Notification\,Window$, so the CA start to chase the $Keep\,Moving$ agents as we introduce from T_{notify_i+2}.

An example of how the agents and the CA move in chordal ring $C_n(1, 2, 7, 11)$is shown in 5.3. For our convenience, in some case, we consider the chordal ring as arranged in rows of d_k where the last node of a row is connected to the first node of the following row and the last node is connected to the first. Depending on the size of the chordal, the last row could be incomplete. So in this matrix, moving down a column corresponding to using the longest chord d_k. In the matrix, we also mark the number of nodes.

0	1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20	21
22	23	24	25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	V	41	42	43
44	45	46	47	48	49	50	51	52	53	54
55	56	57	58	59	60	61	62	63	64	65

Figure 5.3: Arrangement of agent at T_{move_2} when the BV is triggered

Yellow nodes are connected to the original BV but guarded by agents while the grey nodes are the new formed BVs. The node marked V is the original BV but now is clean. Agent residing in node 29 receives clone from chord d_k so it knows it is the CA During the notifying time, agents residing in nodes 33, 38, 39 notify agents residing in nodes 36, 31, 30 respectively following the $Three\,Jump\,Notifying\,Technique$ while the CA moves to 28, 39, 50 and finally 61 following the route in scenario 3.

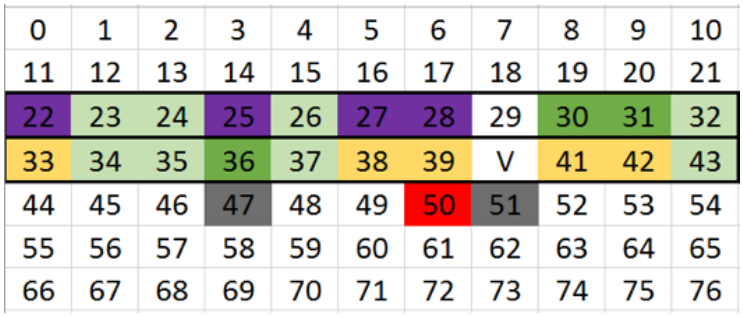


Figure 5.4: Agents roles after $Three\,Jump\,Notifying\,Technique$ and choosing CA. (for convenience, we donate the CA by a red spot, more specifically, node 50 is where CA resides)

Agents in purple nodes would be notified at T_{move_3} and move back. Agents in light green nodes are the $Keep\,Moving$ agents while agent in dark green nodes are informed to stop in $Three\,Jump\,Notifying\,Technique$. In the meantime, the CA moves to node 28, 39, 50, and finally 61. It is obvious that the CA can reach its destination before T_{move_4}, so it waits until T_{notify_4} to start its notifying phase.

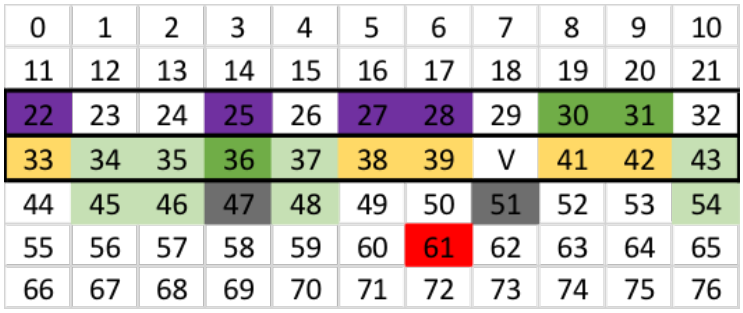


Figure 5.5: Arrangement of agent at T_{move_3}. The CA has arrived its destination)

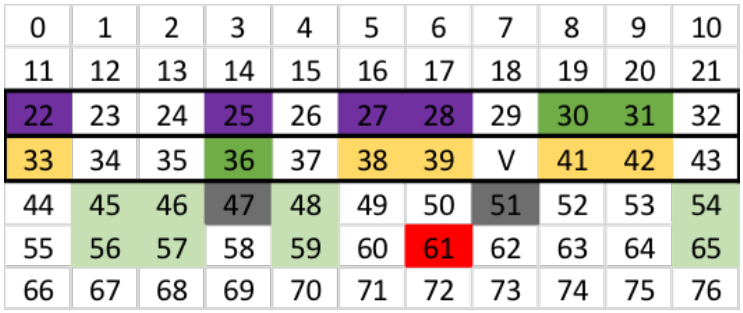


Figure 5.6: Arrangement of agents at T_{move_4}. The CA starts its notice phase

In notifying phase, CA starts to notify other $Keep\,Moving$ agents. First, it computes the $Notification\,Window$ which is from node 55 to node 65. Note that the $Notification\,Window$ would move along chord d_k at every T_{move}. It moves to node 62 at T_{notify_4}, node 63 at $T_{notify\_4'}$, node 64 at $T_{notify\_4''}$ and to node 75 at T_{move_5}.

0	1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20	21
22	23	24	25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	V	41	42	43
44	45	46	47	48	49	50	51	52	53	54
55	56	57	58	59	60	61	62	63	64	65
66	67	68	69	70	71	72	73	74	75	76
77	78	79	80	81	82	83	84	85	86	87

Figure 5.7: Arrangement of agents at T_{move_5}.

Again, in the notifying phase, CA moves to node 76 at T_{notify_5}. We can see that it encounters agent residing in node 76, so CA informed it the $T_{trigger}$ which is T_{move_2}. Agent residing in node 76 should remember T_{notify_now} which is T_{notify_5} and wait until next T_{move} to inform agent ($Following\,Agent$) who resides in node 65 now but would move to node 76 next T_{move}. After encountering its $Following\,Agent$, it informs it to move back along chord d_k for $T_{move\_now} - T_{trigger} + 1$ times which is $T_{move\_5} - T_{move\_2} + 1$ times while itself moves for T_{move_5}- T_{move_2} times. At $T_{T_{notify\_5'}}$ when the CA arrives at node 77, it knows that it just pass its $Ending\,Flag$ so it moves along the longest chord anticlockwise to its $Beginning\,Flag$ at $T_{notify\_5''}$.

0	1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20	21
22	23	24	25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	V	41	42	43
44	45	46	47	48	49	50	51	52	53	54
55	56	57	58	59	60	61	62	63	64	65
66	67	68	69	70	71	72	73	74	75	76
77	78	79	80	81	82	83	84	85	86	87

Figure 5.8: Arrangement of agents at $T_{move''_5}$.

0	1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20	21
22	23	24	25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	V	41	42	43
44	45	46	47	48	49	50	51	52	53	54
55	56	57	58	59	60	61	62	63	64	65
66	67	68	69	70	71	72	73	74	75	76
77	78	79	80	81	82	83	84	85	86	87
88	89	90	91	92	93	94	95	96	97	98
99	100	101	102	103	104	105	106	107	108	109

Figure 5.9: Arrangement of agents at $T_{move''_7}$.

We could know that the CA moves back to its relative original position at $T_{notify\_7''}$. It would wait until next T_{move} to inform its *Following Agent* of $T_{trigger}$ and moves back with it.

5.3.2 Overview of the Elimination

After all the agents move back to where they are when the BV is triggered, we start the *Surrounding and Elimination*. We can destroy the BVs sequentially which is simple to execute but might in some case cost more time. In this way, because all the agents are aware of the positions of the new-formed BVs, so every time at most $d-1$ agents are sent to surround one of the BV and one agent is sent to destroy the BV and move on to

decontaminate the second BV in a same way. Actually, this elimination strategy is the same as that in [1]. Or if we care most the execution time , we can destroy the BVs at one time. First we need to guard all the neighbouring nodes of the new formed BVs. In order to avoid collision and efficiently leverage the agents, we allocate different Destination Tables to all the agents in the array to inform them where should they should move in different situations (e.g., when the first agent in the exploring team is destroyed, then every agent except the first agent have a distinct destination, when the second agent is destroyed, then every agent except that agent destroyed have a distinct destination. More specifically, for a Chord Ring with half degree d, every agent in the array carries a $Destination\,Table$ with $d-1$ destinations. If we need more agents, then we will give their $Destination\,Table$ to the last agent in the shadowing group, when the elimination begins, it clones enough number of agents and give the $Destination\,Table$ to them. Before moving to its destination, the agent computes the shortest route from its own position to its destination using Dijkstra Algorithm. There are two kinds of agent in the Elimination phase: surrounding agents who are responsible for guarding the neighbouring nodes of the BVs and destroying agents who move to the BVs after all the neighbouring nodes are guarded. We want the BVs to be destroyed at one time, so it is important that the destroying agents move to the BVs at the same time and only after all the neighboring nodes are guarded by agents. In fact, if the destroying agents know the longest time $t_{longest}$ to move to the destination taken by all the agents (including the destroying agents and the surrounding agents), then they move to the last node prior to the destination and wait until $t_{longest}$ to move to the BVs together. So in the $Destination\,Tables$ for the destroying agents, we also add an item which is the $t_{longest}$. Now we introduce how to compute the shortest routes and how we design the $Destination\,Tables$. Note that we design $Destination\,Tables$ for all the agents and allocate them to the agents before the exploring phase begins.

5.3.3 Destination Table and Elimination

Supposing there are some BVs and agents in the chordal ring, it is obvious that the BV nodes are in the clockwise side of the agents. In order to use Dijkstra, first we need to map the chordal ring with BVs into a graph. We include nodes from the node containing the first agent to the node which is d_k away from the last BV node, then delete the chords from the BV nodes to build the graph where we run Dijkstra Algorithm. Here is an example how we built the graph for running Dijkstra Algorithm. Below we show the situation when the third agent in the exploring group is destroyed by the BV (see 5.10). Only the chords of the original BV node are shown.

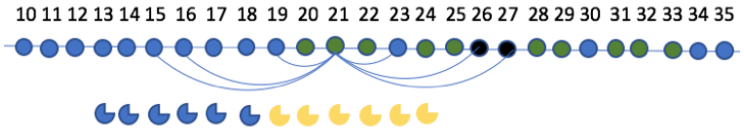


Figure 5.10: Situation when the third agent in the exploring group is destroyed

The black node is the BV node while the green nodes need to be guarded. So in this case, we need 12 agents (10 surrounding agents and 2 destroying agents). We add nodes from 13 to 33 with their chords within this area and delete chords connected with the BV nodes to get the graph where we use Dijkstra Algorithm. Below is the graph we build. (see 5.11) For convenience, we show the all the nodes we included and the chords we delete.

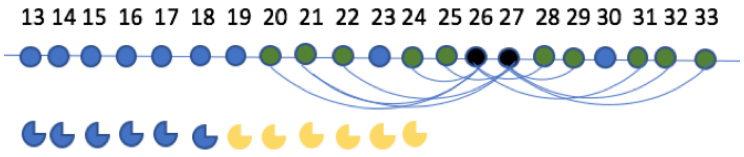


Figure 5.11: The graph we build for Dijkstra Algorithm

Using the graph and Dijkstra Algorithm, we compute the routes from every agent to every node. Then we use enumeration to choose an allocation of every agents's destination satisfying:

- 1) the maximum length of the route should be minimum.
- 2) after the allocation, in every needed position there should be exactly one agent.

After we get the optimal allocation, we record every agent?s distinct destination and the position of the third agent in the exploring group in their $Destination\,Table$. Also, we add the length of the longest chord to every destroying agent. More specifically, here we talk about the situation when the third agent in the exploring group is destroyed. For every agent, the information we compute would be in the third part of its $Destination\,Table$ recording the position of the third exploring agent (for example, it connects this agent through chord x), the destination it should move if the third exploring is destroyed. For a destroying agent, there should be another item recording the length of the longest route in this part. Above we introduce how to design one part of $Destination\,Table$ of an agent, for every agent in chordal ring, it should hold a $Destination\,Table$ of $d-1$ parts, and every part contains 2 items (for surrounding agent) or 3 items (for destroying agent). After the one of the agent is destroyed, the agent can check their $Destination\,Table$ to get the information of their destination. Then using Dijkstra Algorithm they can compute the shortest route separately and starts to move.

5.4 Analysis and Comparing

5.4.1 Theorem and Proof

Theorem 1. *If the structure of a chordal ring is fixed, no mater where is the BV, the number of $Keep\,Moving$ agent is a constant.*

Proof. Define: Going column, the column in which the agents keep going when an explorer encounters a virus.

Define: Stop column, the column in which the agents stop moving when an explorer

encounters a virus. We assume that the structure of the ring is $C_n(1, d_2, d_3, \ldots, d_k)$, and there are d columns in the matrix which are $M = c_0, c_1, \ldots, c_{d-1}$

We assume the number of node where the original BV is V(P_V), then it should in the column $c_{v \bmod d}$.

A column is a stop column if and only if $\exists i, j \in [1, d]$, so that $(V + r_i) \bmod d = S$ or $(V - r_i) \bmod d = S$.

If the explorer encounters the virus at the position P_{V+a} instead of P_V. Since $(V + r_i + a) \bmod d = (S + a) \bmod d$ or $(V - r_i + a) \bmod d = (S + a) \bmod d$, the column $c_{(S+a) \bmod d}$ should be a stop column.

In another word, if c_S is a stop column when virus position is P_V, then $c_{(S+a) \bmod d}$ is a stop column when the virus position is P_{V+a}.

We assume we have two different stop columns c_x and c_y , when the black virus's position is P_V. So when the black virus's position is P_{V+a}, the columns are $c_{(x+a) \bmod d}$ and $c_{(y+a) \bmod d}$. It is obviously that $\forall a$, if $x \neq y$, then $(x + a) \bmod d \neq (y + a) \bmod d$. That means we also have two different stop columns. So that the number of stop column does not decrease, which means that given a fixed chordal ring C and the number of agent keeping moving when the BV is in V (V can be any position), then the number of agent keeping moving when the BV is in other position does not decrease.

Let us assume that the number of agents keeping moving in different case when the longest chord remains the same is different and donates the minimum number of going columns among them by $N_{minimum}$ while the maximum number of going columns among them by $N_{maximum}$, then according to our conclusion $N_{minimum} \geq N_{maximum}$, which means that $N_{minimum} = N_{maximum}$. In another word, the number of agents keeping moving is a constant when the structure of the chordal ring is fixed. □

5.4.2 Analysis and Comparing

Time cost analysis and comparing We only consider the situation when n (the number of nodes of the chordal ring) is much larger than d_k, and since the time cost in the elimination phase is $O(1)$. Finally, we compute the TWT of both protocols to present a more fair comparison. Let us assume that the total number of moves is M, then the worst case costing the most time is when the BV is located at any nodes with number from $n-d_k+1$ to $n-1$ and the first agent (anticlockwise) is in the node with the number $n-d_k$. In this case, it cost $M = \left\lfloor \frac{n}{d_k} - 1 \right\rfloor$ moves and $4M$ units of time ($4M = 4 \times \left\lfloor \frac{n}{d_k} - 1 \right\rfloor$) to finish the exploring phase. In [1], she give the number of move in three case.

- 1) In double loops the upper bound of moves is $4n-7$.
- 2) In the triple loops, she discusses two classes of chordal ring: $C_n(1,p,k)$ and $C_n(1,d_k-1,d_k)$.In the first case, the number of moves needed is $5n-6d_k+22$ while in the second case, a maximum of $5n-7d_k+22$ moves are needed.
- 3) In the consecutive-chordal rings, a maximum of $(d_k+2)n-2d_k-3$ moves are needed.

Since in the sequential strategy, agents do not need to wait so the time cost is equal to the number of moves. And it is obvious that our protocol is much faster than the sequential strategy. But since we use much more agents, so in order to gain a fair comparison, now we compute TWT of both protocol. In the exploring phase, we use $2d_k$ agents, so the TWT of our protocol is $8n-8d_k$. In the exploring phase of the sequential strategy, it need at least 2 agent to explore and some other shadow agents to guard the explored nodes but the number of shadow depends on the structure of the chordal ring so now we ignore them. Now we compute TWT of the sequential strategy.

- 1)In double loops the upper TWT is $8n-14$

- 2)The TWT in chordal ring $C_n(1, p, d_k)$ is $10n-6d_k+44$ and in chordal ring $C_n(1, d_k-1, d_k)$ is $10n - 14d_k + 44$.

- 3)The TWT in consecutive-chordal rings is $2(d_k + 2)n - 4d_k - 6$.

It is obvious that when $d_k \geq 2$, our protocol is faster in first case; when $d_k \leq \frac{1}{3}n + 7$,our protocol is faster in the second case (both $C_n(1, p, k)$ and $C_n(1, d_k - 1, d_k)$); when $d > 2 - \frac{1}{n+2}$, our protocol is faster in the third case.

Calamity Analysis Casualty is the number of agents destroyed by the BV. In chordal ring $C_n(1, d_2, \ldots, d_k)$, the worst case is that the first agent in the exploring group is destroyed by a BV and the clones of it spread to all its neighbouring nodes. The casualties in this case are $d_k + 1$ because another d_k nodes are guarded by agents while in sequential case, the casualties are $2d_k$. So in terms of casualty, our protocol is better than the sequential strategy.

References

[1] Modhawi Alotaibi. Black virus disinfection in chrodal rings. Master's thesis, University of Ottawa, 2014.

[2] L. Barri'ere, P. Flocchini, F. V. Fomin, P. Fraignaud, N. Nisse, N. Santoro, and D. M. Thilikos. Connected graph searching. *Information and Computation*, pages 1–16, 2012.

[3] L. Barri'ere, P. Flocchini, P. Fraignaud, and N. Santoro. Capture of an intruder by mobile agents. In *14th Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 200–209, 2002.

[4] L. Blin, P. Fraignaud, N. Nisse, and S. Vial. Distributed chasing of network intruders. *Theoretical Computer Science*, 2006.

[5] J. Chalopin, S. Das, A. Labourel, and E. Markou. Tight bounds for black hole search with scattered agents in synchronous rings. *Theoretical Computer Science*, pages 70–85, 2013.

[6] C. Cooper, R. Klasing, and T. Radzik. Locating and repairing faults in a network with mobile agents. *Theoretical Computer Science*, 2010.

[7] C. Cooper and R. Tomasz. Searching for black-hole faults in a network using multiple agents. *In Proceedings of 10th International Conference on Principles of Distributed Systems*, 2006.

[8] J. Czyzowicz, S. Dobrevn, R. Kr alovi c, S. Mikl ık, and D. Pardubska. Black hole search in directed graphs. In *In Proceedings of the 17th International Colloquium on Structural Information and Communication Complexity*, pages 182–194, 2010.

[9] J. Czyzowicz, D. R. Kowalski, E. Markou, and A. Pelc. Searching for a black hole in tree networks. In *In Proceedings of the 8th International Conference on Principles of Distributed Systems*, pages 67–80, 2004.

[10] D. Dereniowski. Connected searching of weighted trees. *Theoretical Computer Science*, pages 5700–5713, 2011.

[11] P. Flocchini, M. Huang, and F. Luccio. Decontaminating chordal rings and tori using mobile agents. *International Journal of Foundations of Computer Science*, pages 547–563, 2007.

[12] P. Flocchini, M. Huang, and F. Luccio. Decontamination of hypercubes by mobile agents. *Networks*, 2008.

[13] P. Flocchini, M. J. Huang, , and F. L. Luccio. Decontamination of hypercubes by mobile agents. *Networks*, pages 167–178, 2008.

[14] P. Flocchini, F.L. Luccio, and L. X. Song. Size optimal strategies for capturing an intruder in mesh networks. *International Conference on Communications in Computing (CIC)*, pages 200–206, 2005.

[15] P. Flocchini, P. Mason M. Kellett, and N. Santoro. Map construction and exploration by mobile agents scattered in a dangerous network. In *In Proceedings of the 23rd IEEE International Symposium on Parallel and Distributed Processing*, pages 1–10, 2009.

[16] F.V.Fomin and D.M.Thilikos. An annotated bibliography on guarabteed graph searching. *Theoretical computer science*, pages 236–245, 2008.

[17] D. Ilcinkas, N. Nisse, and D. Soguet. The cost of monotonicity in distributed graph searching. *Distributed Computing*, pages 117–127, 2009.

[18] J.Cai, P.Flocchini, and N.Santoro. Decontaminating a network from a black virus. *International Journal of Networking and Coputing*, pages 151–173, 2014.

[19] J.Cai, P.Flocchini, and N.Santoro. Distributed black virus decontamination and rooted acyclic orientations. In *CIT/IUCC/DASC/PICom*. IEEE, 2015.

[20] J.Chalopin, S.Das, A.Labourel, and E.Markou. Black hole search with finite automata scattered in a synchronous torus. In *25th International Symposium on Distributed Computing(DISC)*, pages 432–446, 2011.

[21] J.Chalopin, S.Das, A.Labourel, and E.Markou. Tight bounds for black hole search with scattered agents in synchronous ring. *Theoretical Computer Science*, pages 70–85, 2013.

[22] A. Kosowski, A. Navarra, and C. M. Pinotti. Synchronous black hole search in directed graphs. *Theoretical Computer Science*, 2011.

[23] F. Luccio, L. Pagli, and N. Santoro. Network decontamination in presence of local immunity. *International Journal of Foundations of Computer Science*, pages 457–474, 2007.

[24] P.Flocchini. Contamination and decontamination in majority-based systems. *Journal of Cellular Automata*, pages 183–200, 2009.

[25] P.Flocchini and N.Santoro. *Distributed Security Algorithms for Mobile Agents*. Mobile Agents in Networking and Distributed Computing, 2012.

[26] S.Das, P.Flocchini, A.Nayak, and N.Santoro. Map constructuction of unknown graphs by multiple agents. *Theoretical Computer Science*, 385(1-3):34–48, 2007.

[27] S.Das, P.Flocchini, and N.Santoro. Mobile search for a black hole in an anonumous ring. *Algorithmica*, pages 67–90, 2007.

[28] S.Dobrev, P.Flocchini, G.Prencipe, and N.Santoro. Searching for a black hole in arbitary networks:optimal mobile agents protocols. *Distributed Computing*, pages 1–18, 2006.

[29] S.Dobrev, P.Flocchini, R.Kralovic, P.Ruzicka, G.Prencipe, and N.Santoro. Black hole search in common interconnection networks. *Networks*, pages 61–71, 2006.